

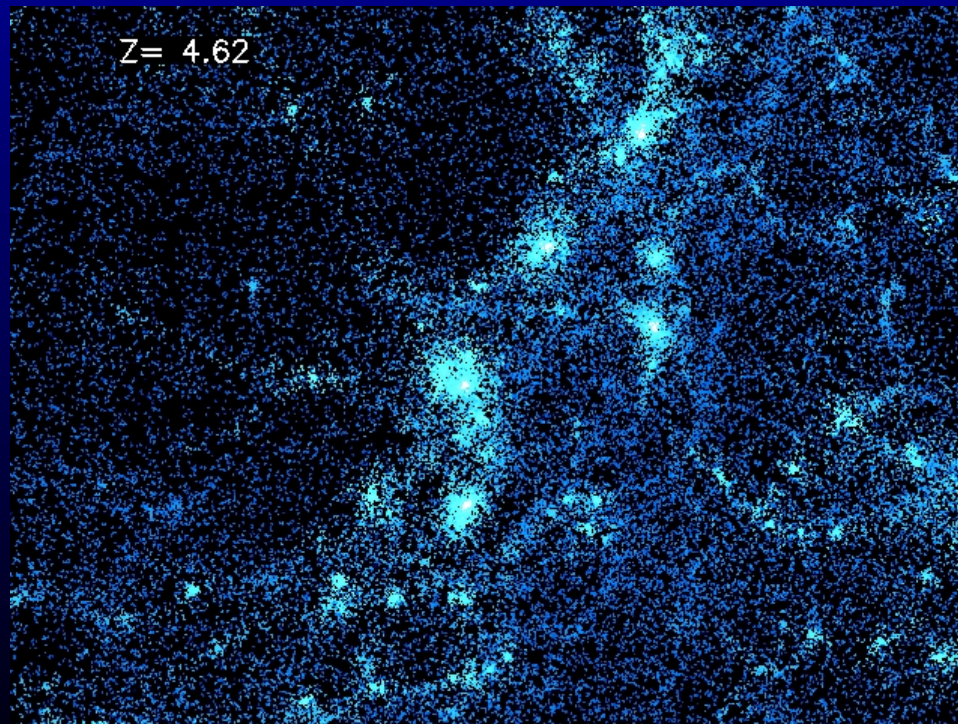
Writing a PM code

Andrey Kravtsov

`andrey@oddjob.uchicago.edu`

`astro.uchicago.edu/~andrey/talks/PM/pmpotsdam.pdf`

July 23, 2006



Cosmological simulations: the problem

- The Universe is thought to be composed of dark matter and normal matter (the baryons), photons, neutrinos, etc.
- There should be $\sim 10^{80 \div 90}$ particles of dark matter and baryons in the observable Universe.
- In our Galaxy, the Milky Way, alone there are $\sim 10^{68}$ baryonic particles.
- Simulations, of course, cannot follow evolution of all of these particles.
- The evolution of a large number of particles of a given type can be described as evolution of a continuous field, characterized by the distribution function $f(\mathbf{x}, \dot{\mathbf{x}}, t)$ — the density of particles in phase space.

Cosmological simulations: equations I

- The evolution of f_i is described by the Boltzmann equation:

$$\frac{\partial f_i}{\partial t} + \dot{\mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}} + \ddot{\mathbf{x}} \frac{\partial f_i}{\partial \dot{\mathbf{x}}} = C[f_i, f_j];$$

- $C[f_i, f_j]$ is the collision integral, describing possible particle collisions, which can scatter particles in and out of a phase-space volume. The form of C depends on the specific properties of particles in the system (i.e., how they scatter, whether they can be created or annihilated, etc.)
- In general, a system of 6D Boltzmann equations (equations for each of the relevant components, i , of the Universe) needs to be solved.
- This is indeed done in codes describing evolution of the primordial matter and linear perturbations to solve for the initial power spectrum and properties of the CMB temperature anisotropies. In this case, the matter distribution is close to uniform and no detailed information on the 3D distribution of matter is needed. The dimensionality of the problem can thus be reduced.

Cosmological simulations: equations II

- When solving for nonlinear structure formation, however, the full 3D spatial information is needed.
- If dark matter is collisionless or nearly collisionless, $C[f] = 0$, resulting in a simpler, *collisionless Boltzmann equation*, a.k.a. *the Vlasov equation*. However, this “simpler” equation is still a 6D ODE, if we want to describe evolution of the matter in 3D space. High-dimensionality of the equation severely limits the resolution of simulations due to the large number of elements required to cover the 6D space.
- The solution of the Vlasov equation can be represented in terms of characteristic equations, which look like equations of particle motion. Characteristic equations describe lines in phase space along which the distribution function is constant.
- A complete set of characteristic equations is equivalent to the Vlasov equation. But if a representative subsample of characteristics can be followed, we can get the approximate evolution of the system.

Cosmological simulations: N -body approach

- This is the main idea behind the particle N -body approach in plasma and cosmological simulations.
- In N -body approach, the initial phase space is split into small domains, each domain is represented by a particle, and particles are evolved self-consistently using equations of particle motion:

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}, \quad \frac{d\mathbf{u}}{dt} = -\nabla\Phi$$

with potential given by the Poisson equation:

$$\nabla^2\Phi = 4\pi G\rho_{\text{tot}}.$$

- N -body approach is effectively a Monte Carlo technique for random sampling of the characteristics.¹

¹Monte Carlo techniques are very often used in the otherwise intractable high-dimensional problems, like evaluating many-dimensional integrals or likelihood and parameter estimation in many-parameter problems. Markov Chain Monte Carlo (MCMC) methods are used in such problems.

Cosmological simulations: particle initial conditions

- If we deal with cold dark matter, initial particle velocities are due solely to gravity and, in the Zeldovich approximation, depend only on particle positions. In this case only 3D space needs to be discretized.
- For other type of dark matter (hot, warm, etc.), their initial velocity distribution must also be sampled.
- The problem of initial discretization is the problem of cosmological initial conditions, covered last week. For additional information see:

Anatoly Klypin's PM code

Ed Bertschinger's GRAFIC2 package: a public cosmological IC code

Grocce et al. 2006, (astro-ph/0606505)

- the number of particles and the size of simulation volume are dictated by the available computational resources and the needs of particular problem.

Cosmological simulations: particle initial conditions

“It seems probably to me that God in the beginning formed matter in solid, massy, hard, impenetrable, movable particles, of such sizes and figures, and with such other properties, and in such proportion to space, as most conduced to the end for which he formed them.”

- Isaac Newton, “Opticks”

Cosmological simulations: equations III: gasdynamics

- In a fully collisional particle systems with properties of an ideal gas, the Boltzmann equation can be simplified and reduced to 3D ODEs describing ideal compressible gas.
- This is done by taking the three first moments of the Boltzmann equations — i.e., multiplying the equation by 1, \dot{x} , and \dot{x}^2 and integrating over velocity space).

Cosmological simulations: equations III: gasdynamics

- These moments give the continuity, Euler, and energy equations of hydrodynamics² (e.g., *F. Shu "Gasdynamics"*):

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{u} = 0,$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla \Phi - \frac{\nabla P}{\rho},$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P)\mathbf{u}] = -\rho \mathbf{u} \cdot \nabla \Phi.$$

- The equations above are closed with the equation of state, which is usually assumed to be that of an ideal gas:

$$\varepsilon = \frac{1}{\gamma - 1} \frac{P}{\rho},$$

or to have some other simple form (i.e., polytropic EoS: $P = K\rho^\gamma$)

²Equations below do not take into account additional processes commonly included in simulations: heating and cooling of gas, star formation, etc. These are included as sink and source terms in the appropriate equations. We will consider such processes on Friday.

Cosmological simulations: equations summary

- Equations of particle motion, the Poisson equation, the equations of gasdynamics above, along with the equation of state of the gas, form the set of basic equations solved in cosmological simulations:

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}, \quad \frac{d\mathbf{u}}{dt} = -\nabla\Phi$$

$$\nabla^2\Phi = 4\pi G\rho_{\text{tot}}.$$

$$\frac{\partial\rho}{\partial t} + \nabla\rho\mathbf{u} = 0,$$

$$\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla\Phi - \frac{\nabla P}{\rho},$$

$$\frac{\partial E}{\partial t} + \nabla \cdot [(E + P)\mathbf{u}] = -\rho\mathbf{u} \cdot \nabla\Phi,$$

$$\varepsilon = f(\rho, P).$$

- Baryonic gas and dark matter are coupled by gravity, as they are both influenced by the same gravitational potential and they both contribute to the density in the Poisson equation.
- Note that all of the equations we solve to describe the evolution of matter in the universe have their usual simple non-relativistic form. This is because most commonly the matter components relevant for late evolution move with non-relativistic velocities.

Cosmological simulations: dissipationless case

- For many problems it is warranted to ignore the baryonic effects and solve only evolution of dissipationless component (dark matter, stars).
- In this case, only equations of particle motion and the Poisson equation are solved.
- Very generally, each evolution time step of a dissipationless simulation consists of the following two main parts:
 - *gravity solver*: solve the Poisson equation and calculate particle accelerations.
 - *time integrator*: update particle positions and velocities.
- different codes differ in how these two parts are implemented.
- Today we will consider the Particle-Mesh (PM) technique for solving the Poisson equation and integrating particle trajectories.

Particle-Mesh (PM) technique: a brief history

- PM was invented in the 1950's at LANL for simulations of compressible fluid flows
- However, the first wide-spread application was for collisionless plasma simulations (for which PM was reinvented in the 1960's and popularized by Hockney and collaborators)
- In the late 70's PM was first applied for 3D cosmological simulations. The method and its descendants (such as P^3M , AP^3M , and ART) have been used in most cosmological simulations ever since ³.

³In the 1990's Tree codes have also enjoyed increasing popularity

Particle-Mesh (PM) technique: modern relevance

- The popularity of PM in cosmology is due to
 - ★ its relative algorithmic simplicity, speed (the running time scales as $\propto O(N_p) + O(N_c \ln N_c)$, where N_p is the number of particles and N_c is the number of grid cells; and low storage requirements;
 - ★ natural incorporation of periodic boundary conditions;
- Today the PM technique is still very much relevant because
 - ★ Simplicity allows for efficient parallelization → and thus simulation on largest distributed architectures;
 - ★ low storage requirements allow simulation of a very large numbers of particles with relatively low resolution, suitable for some problems (e.g., halo mass function). Papers based on pure PM simulations still appear.
 - ★ PM technique is used to solve for the large-scale potential and forces in many modern high-resolution methods (e.g., **AP³M**, ART, **TreePM⁴**).

⁴now implemented in the **Gadget-2** code

Model equations

PM code solves the cosmological Poisson equation (Peebles 1980, LSS):

$$\nabla^2\Phi = 4\pi G\rho_{\text{tot}} - \Lambda,$$

and equations of motion of particles

$$\frac{d\mathbf{r}}{dt} = \mathbf{u}; \quad \frac{d\mathbf{u}}{dt} = -\nabla\Phi,$$

note that all variables are defined in proper coordinates and all spatial derivatives are also taken with respect to these coordinates.

However, it is convenient to re-write the equations in *comoving* variables and make them dimensionless by choosing suitable units (I will denote variables in code units with a tilde $\tilde{}$). In the following, I will adopt the **variables and units used in Anatoly Klypin's PM code**. This is just an example. Feel free to choose your own - just make sure you are consistent!

$$\tilde{\mathbf{x}} \equiv a^{-1} \frac{\mathbf{r}}{r_0}, \quad \tilde{\mathbf{p}} \equiv a \frac{\mathbf{v}}{v_0}, \quad \tilde{\phi} \equiv \frac{\phi}{\phi_0}, \quad \tilde{\rho} = a^3 \frac{\rho}{\rho_0},$$

where \mathbf{x} is the comoving coordinates, $\mathbf{v} = \mathbf{u} - H\mathbf{r} = a\dot{\mathbf{x}}$ is the *peculiar velocity*⁵ and ϕ is the *peculiar potential* defined as (Peebles 1980, p. 42)

$$\phi = \Phi + 1/2 a\ddot{a} (r/a)^2 = \Phi + \frac{H_0^2}{2} \left(\Omega_{\Lambda,0} - \frac{1}{2} a^{-3} \Omega_{m,0} \right) r^2,$$

where

$$\Omega_{m,0} = \frac{8\pi G\rho_0}{3H_0^2}; \quad \Omega_{\Lambda,0} = \frac{\Lambda}{3H_0^2}.$$

⁵ $p \propto av$ is called momentum, this choice of variable allows us to get rid of the annoying (\dot{a}/a) terms in equations.

The quantities with subscript zero are the units in which corresponding physical variables are measured. The unit of length, r_0 , is an arbitrary scale. I will chose r_0 to be the size of the PM grid cell:

$$r_0 = \frac{L_{\text{box}}}{N_g}; \quad N_g^3 = \text{total number of grid cells}$$

the rest of the units are defined as

$$t_0 \equiv H_0^{-1},$$

$$v_0 \equiv \frac{r_0}{t_0},$$

$$\rho_0 \equiv \frac{3H_0^2}{8\pi G} \Omega_{m,0},$$

$$\phi_0 \equiv \frac{r_0^2}{t_0^2} = v_0^2.$$

It is also convenient to choose the expansion factor as time variable (using expressions for the Hubble constant: $\dot{a} = aH(a)$). For this choice of variables, the Poisson equation and equations of motion can be re-written as

$$\nabla^2 \phi = 4\pi G \Omega_{m,0} \rho_{crit,0} a^{-1} \delta, \quad \delta = \frac{\rho - \bar{\rho}}{\bar{\rho}},$$

$$\frac{d\mathbf{p}}{da} = -\frac{\nabla\phi}{\dot{a}}, \quad \frac{d\mathbf{x}}{da} = \frac{\mathbf{p}}{\dot{a}a^2}.$$

where δ is the overdensity in comoving coordinates and \dot{a} is

$$\dot{a} = H_0 a^{-1/2} \sqrt{\Omega_{m,0} + \Omega_{k,0}a + \Omega_{\Lambda,0}a^3}; \quad \Omega_{m,0} + \Omega_{\Lambda,0} + \Omega_{k,0} = 1$$

In dimensionless variables the equations are:

$$\tilde{\nabla}^2 \tilde{\phi} = \frac{3\Omega_0}{2a} \tilde{\delta},$$

$$\frac{d\tilde{\mathbf{p}}}{da} = -f(a)\tilde{\nabla}\tilde{\phi}, \quad \frac{d\tilde{\mathbf{x}}}{da} = f(a)\frac{\tilde{\mathbf{p}}}{a^2}.$$

where $\tilde{\delta} = \tilde{\rho} - 1$ and

$$f(a) \equiv H_0/\dot{a} = [a^{-1} (\Omega_{m,0} + \Omega_{k,0}a + \Omega_{\Lambda,0}a^3)]^{-1/2}.$$

These equations are used in the three main steps of a PM code:

- Solve the Poisson equation using the density field estimated with current particle positions.
- Advance momenta using the new potential.
- Update particle positions using new momenta.

Data structures

For a PM code with the second-order accurate time integration we need the minimum of *six real numbers for positions and momenta for each particle* (assuming particles have the same mass) and *one real number for the potential of each grid cell*.

The array for the potential can be shared between density and potential: first use it for density, then replace it with potential when the Poisson equation is solved. However, for simplicity you can start with two grid arrays (for density and potential). Also, you will probably need auxiliary arrays for FFT, depending on which FFT solver you choose to use.

The convenient data structures are 1D or 3D arrays for particles (e.g., six 1D arrays for $x(i)$, $y(i)$, $z(i)$, $vx(i)$, $vy(i)$, $vz(i)$) and 3D arrays for the grid variables (e.g., $\rho(i, j, k)$, $\phi(i, j, k)$).

Run your tests for $(32^3, 64^3)$ or $(64^3, 128^3)$ particles and cells in which case memory requirements should not be an issue.

Density Assignment

The Particle Mesh algorithms assume that particles have certain size, mass, shape, and internal density. This determines the interpolation scheme used to assign densities to grid cells. Let's define the 1D particle shape, $S(x)$, to be mass density at the distance x from the particle for cell size Δx (Hockney & Eastwood 1981). The common choices are

- **Nearest Grid Point (NGP)**: particles are point-like and all of particle's mass is assigned to the single grid cell that contains it:

$$S(x) = \frac{1}{\Delta x} \delta \left(\frac{x}{\Delta x} \right)$$

- **Cloud In Cell (CIC)**: particles are cubes (in 3D) of uniform density and of one grid cell size.

$$S(x) = \frac{1}{\Delta x} \begin{cases} 1, & |x| < \frac{1}{2}\Delta x \\ 0, & \text{otherwise} \end{cases}$$

- **Triangular Shaped Cloud (TSC):**

$$S(x) = \frac{1}{\Delta x} \begin{cases} 1 - |x|/\Delta x, & |x| < \Delta x \\ 0, & \text{otherwise} \end{cases}$$

The fraction of particle's mass assigned to a cell ijk is the shape function averaged over this cell:

$$W(x_p - x_{ijk}) = \int_{x_{ijk} - \Delta x/2}^{x_{ijk} + \Delta x/2} dx' S(x_p - x');$$

$$W(\mathbf{r}_p - \mathbf{r}_{ijk}) = W(x_p - x_{ijk})W(y_p - y_{ijk})W(z_p - z_{ijk});$$

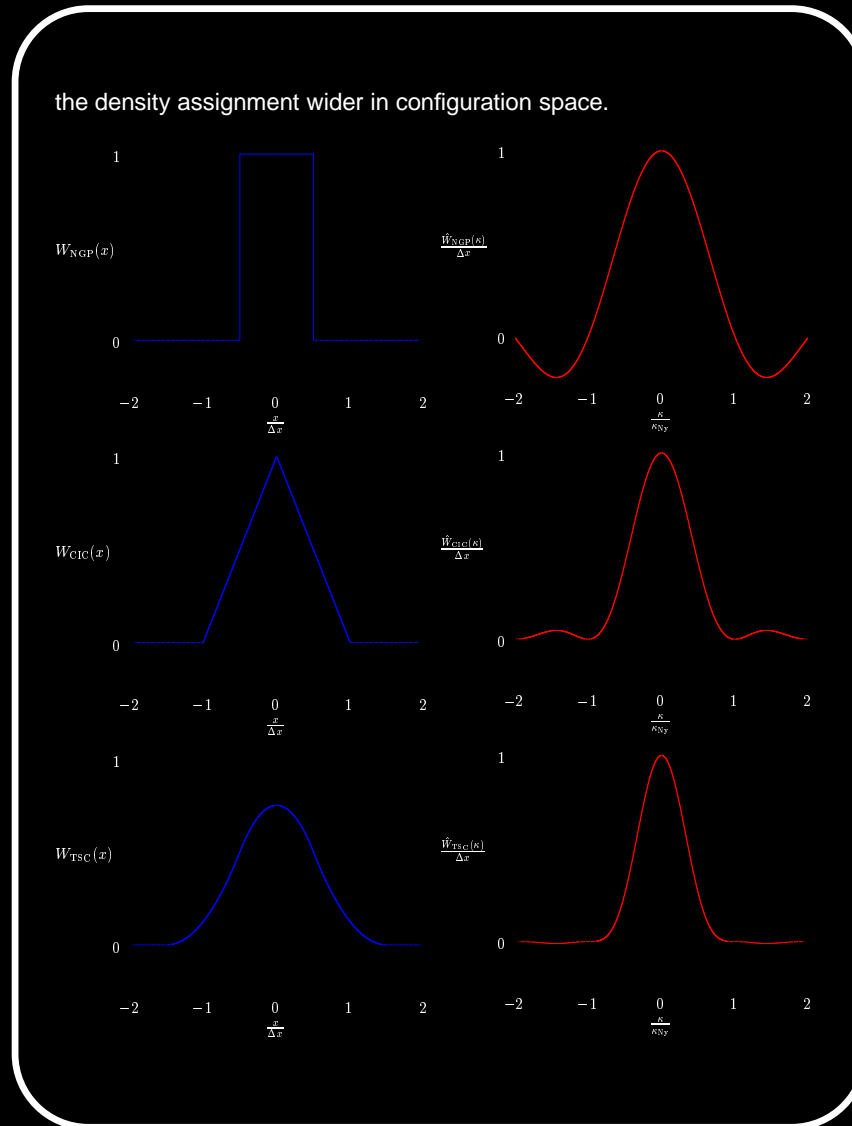
The density in a cell ijk is then

$$\rho_{ijk} = \sum_{p=1}^{N_p} m_p W(\mathbf{r}_p - \mathbf{r}_{ijk})$$

In practice, of course, we loop over particles and assign the density to neighboring cells as opposed to summing over all particles for each cell as the straightforward reading of the above equation would suggest.

PM interpolation kernels in real and Fourier space. Adopted from [PM lectures by M.Gross](#)

3-9



Implementing CIC interpolation

The choice of interpolation scheme is a tradeoff between accuracy and computational expense. The CIC scheme is both relatively cheap and accurate and is most commonly used in PM codes. I will now describe how to implement it (you can start coding with the simpler NGP scheme and upgrade it to CIC later when your code is tested).

Consider the density assignment for a particle with coordinates $\{x_p, y_p, z_p\}$. The cell containing the particle will have indices of

$$i = [x_p]; \quad j = [y_p]; \quad k = [z_p],$$

where $[x]$ is the integer floor function (equivalent to Fortran's `int`). Let's assume that cell centers are at $\{x_c, y_c, z_c\} = \{i, j, k\}$ (note that in internal units cell size $\Delta x = 1$, this will be implicitly assumed below). This is a matter of convention, but once chosen the convention should be consistent throughout the code.

In the CIC scheme, particle $\{x_p, y_p, z_p\}$ may contribute to densities in the parent cell (i, j, k) and seven neighboring cells⁶. Let's define

$$d_x = x_p - x_c; \quad d_y = y_p - y_c; \quad d_z = z_p - z_c;$$

$$t_x = 1 - d_x; \quad t_y = 1 - d_y; \quad t_z = 1 - d_z.$$

Contributions to the eight cells are then linear interpolations in 3D:

$$\rho_{i,j,k} = \rho_{i,j,k} + m_p t_x t_y t_z; \quad \rho_{i+1,j,k} = \rho_{i+1,j,k} + m_p d_x t_y t_z;$$

$$\rho_{i,j+1,k} = \rho_{i,j+1,k} + m_p t_x d_y t_z; \quad \rho_{i+1,j+1,k} = \rho_{i+1,j+1,k} + m_p d_x d_y t_z;$$

$$\rho_{i,j,k+1} = \rho_{i,j,k+1} + m_p t_x t_y d_z; \quad \rho_{i+1,j,k+1} = \rho_{i+1,j,k+1} + m_p d_x t_y d_z;$$

$$\rho_{i,j+1,k+1} = \rho_{i,j+1,k+1} + m_p t_x d_y d_z; \quad \rho_{i+1,j+1,k+1} = \rho_{i+1,j+1,k+1} + m_p d_x d_y d_z;$$

where m_p is particle mass. Doing this for all particles will result in the grid of densities $\rho_{i,j,k}$.

⁶Make sure you enforce periodic boundary conditions: $i = \text{mod}(i, N_{g,1})$, etc. for j and k .

Solving the Poisson equation

With the grid of densities, $\rho_{i,j,k}$, in hand, the code can proceed to solve the discretized⁷ Poisson equation

$$\begin{aligned}\tilde{\nabla}^2 \tilde{\phi} &\approx \tilde{\phi}_{i-1,j,k} + \tilde{\phi}_{i+1,j,k} + \tilde{\phi}_{i,j-1,k} + \tilde{\phi}_{i,j+1,k} + \tilde{\phi}_{i,j,k-1} + \tilde{\phi}_{i,j,k+1} - 6\tilde{\phi}_{i,j,k} = \\ &= \frac{3\Omega_0}{2a}(\tilde{\rho}_{i,j,k} - 1).\end{aligned}$$

The discretization thus results in a large system of linear equations relating unknowns, $\tilde{\phi}_{i,j,k}$, to the known right hand side values. This system can be solved using FFT.

⁷It is customary in PM codes to discretize the Laplacian operator using the 7-point template.

In the Fourier space the Poisson equation is $\tilde{\phi}(\mathbf{k}) = G(\mathbf{k})\tilde{\delta}(\mathbf{k})$, where $G(\mathbf{k})$ is the Green function which for the adopted discretization is given by

$$G(\mathbf{k}) = -\frac{3\Omega_0}{8a} \left[\sin^2 \left(\frac{k_x}{2} \right) + \sin^2 \left(\frac{k_y}{2} \right) + \sin^2 \left(\frac{k_z}{2} \right) \right]^{-1},$$

where $L = N_g$ is the box size in code units and

$$k_x = 2\pi l/L, \quad k_y = 2\pi m/L, \quad k_z = 2\pi n/L, \quad \text{for component } (l, m, n).$$

The singularity at $l = m = n = 0$ should be avoided by setting this component of potential, $\hat{\phi}_{000}$, by hand to zero.

⇒ Given a density field $\tilde{\delta}(\mathbf{r})$ in real space, we can solve for the gravitational potential by

- performing the FFT to get $\tilde{\delta}(\mathbf{k})$,
- multiplying every element in the field by the corresponding value of $G(\mathbf{k})$ to get $\tilde{\phi}(\mathbf{k})$,

$$\hat{\phi}_{lmn} = G(\mathbf{k}_{lmn})\hat{\rho}_{lmn}, \text{ where}$$

$$\hat{f}_{lmn} = (\Delta x)^3 \sum_{i,j,k=0}^{N_g-1} f_{ijk} e^{-i2\pi(il+jm+kn)/N_g},$$

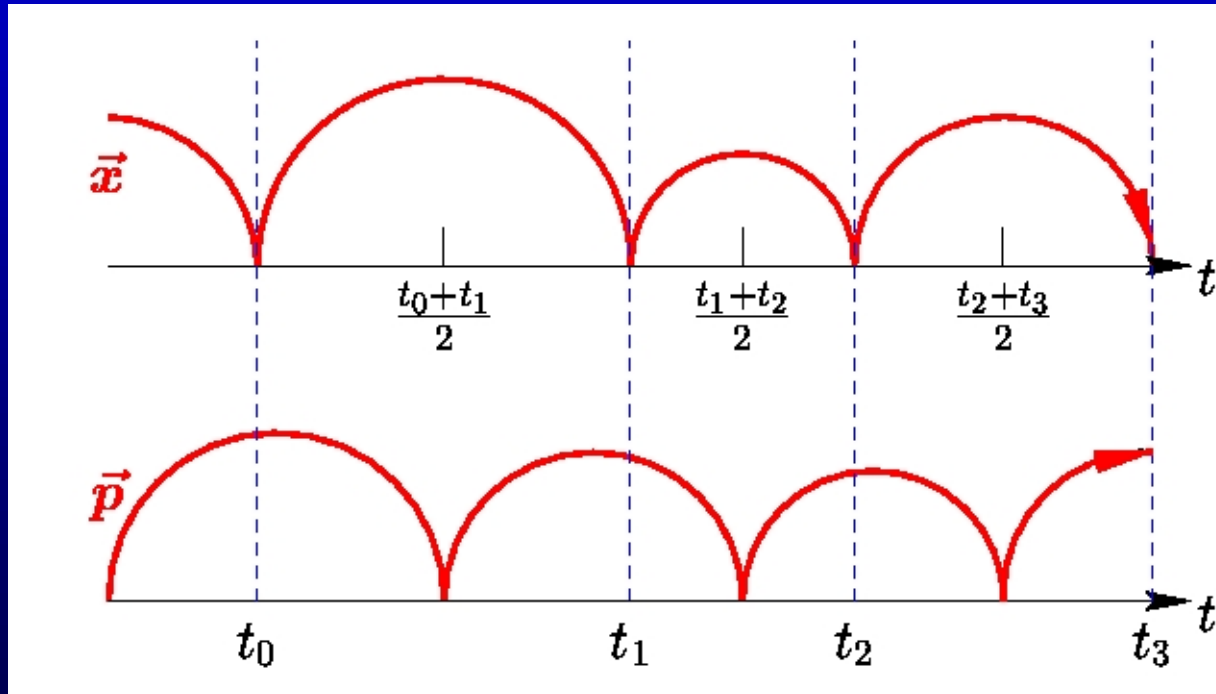
$$f_{ijk} = \frac{1}{L^3} \sum_{l,m,n=0}^{N_g-1} \hat{f}_{lmn} e^{i2\pi(il+jm+kn)/N_g}.$$

- transforming⁸ the result back to real space to get $\tilde{\phi}(\mathbf{r})$ discretized at cell centers.

⁸Be careful about normalization of the FFT. The transforms $\tilde{\delta}(\mathbf{r}) \rightarrow \tilde{\delta}(\mathbf{k})$ and $\tilde{\delta}(\mathbf{k}) \rightarrow \tilde{\delta}(\mathbf{r})$ should recover the original field $\tilde{\delta}(\mathbf{r})$.

Updating particle positions and velocities

Let us assume we are using constant integration step in Δa and the second-order accurate *leapfrog* integration.



Schematic of the leapfrog integration for a variable time step. Note how velocities and particles are staggered in time. (adopted from M.Gross's PM notes).

After n time steps, $a_n = a_i + n\Delta a$, during the step $n + 1$, we should have coordinates $\tilde{\mathbf{x}}_n$ at a_n and momenta $\tilde{\mathbf{p}}_{n-1/2}$ at $a_{n-1/2} = a_n - \Delta a/2$ from the previous step. Assigning density and solving the Poisson equation gives potential $\tilde{\phi}_n$ at a_n . For the assumed variables and units, positions and momenta are updated as follows:

$$\tilde{\mathbf{p}}_{n+1/2} = \tilde{\mathbf{p}}_{n-1/2} + f(a_n)\tilde{\mathbf{g}}_n\Delta a; \quad \tilde{\mathbf{x}}_{n+1} = \tilde{\mathbf{x}}_n + a_{n+1/2}^{-2}f(a_{n+1/2})\tilde{\mathbf{p}}_{n+1/2}\Delta a.$$

Here, $\tilde{\mathbf{g}}_n = -\tilde{\nabla}\tilde{\phi}_n$ is acceleration *at the particle's position*. This acceleration can be obtained by interpolating accelerations from the neighboring cell centers. The latter are given by

$$\tilde{g}_{i,j,k}^x = -(\tilde{\phi}_{i+1,j,k} - \tilde{\phi}_{i-1,j,k})/2, \quad \tilde{g}_{i,j,k}^y = -(\tilde{\phi}_{i,j+1,k} - \tilde{\phi}_{i,j-1,k})/2,$$

$$\tilde{g}_{i,j,k}^z = -(\tilde{\phi}_{i,j,k+1} - \tilde{\phi}_{i,j,k-1})/2.$$

For each component of the acceleration, you should use the same interpolation scheme as during density assignment. For a given particle the acceleration should be interpolated from the cells to which particle contributed density during density assignment step. *This is important!* Consistency in interpolation schemes ensures absence of artificial self-forces and the third Newton's law (Hockney & Eastwood 1981).

For the NGP scheme, acceleration for a particle is just the acceleration of its parent cell $\tilde{g}_{i,j,k}$. For the CIC interpolation, we do the following (indices (i, j, k) here correspond to the parent cell of the particle; see section on density assignment).

$$g_p^x = g_{i,j,k}^x t_x t_y t_z + g_{i+1,j,k}^x d_x t_y t_z + g_{i,j+1,k}^x t_x d_y t_z + g_{i+1,j+1,k}^x d_x d_y t_z + \\ g_{i,j,k+1}^x t_x t_y d_z + g_{i+1,j,k+1}^x d_x t_y d_z + g_{i,j+1,k+1}^x t_x d_y d_z + g_{i+1,j+1,k+1}^x d_x d_y d_z.$$

And the same for g_y^p and g_z^p . $t_{x,y,z}$ and $d_{x,y,z}$ have the same definition as in the CIC density assignment described above.

After updating particle positions and velocities, the code should do some useful I/O and then proceed to step $n + 2$.

Zeldovich approximation

- A first-order (linear) Lagrangian collapse model

[Zeldovich 1970; see also Peacock's (p. 485) and Padmanabhan's (p. 294) textbooks for pedagogical introduction]

- Zeldovich approximation can be expressed by one equation:

$$\mathbf{x}(t) = \mathbf{q} + D_+(t)\mathbf{S}(\mathbf{q}),$$

it may look more familiar if you think about it as

$$\mathbf{x}(t) = \mathbf{x}_0 + \mathbf{v}t; \quad \mathbf{v} = \text{const}$$

where

\mathbf{q} is the initial position of a matter parcel (a particle in N -body simulations) and

$\mathbf{x}(t)$ is its position at time t (both \mathbf{q} and \mathbf{x} are *comoving*).

$D_+(t)$ is the linear growth function [see eq. (29) in [Carroll et al. \(1993\)](#) for a useful fitting formula] and

$\mathbf{S}(\mathbf{q})$ is a *time-independent* displacement vector.

- The evolution of density follows from conservation of mass $\rho(\mathbf{x})d^3\mathbf{x} = \bar{\rho}d^3\mathbf{q}$:

$$\rho(\mathbf{x}, t) = \frac{\bar{\rho}}{\det(\partial x_j / \partial q_i)} = \frac{\bar{\rho}}{\det[\delta_{ij} + D_+(t) \times (\partial S_j / \partial q_i)]}$$

- ZA is exact in 1D until the first crossing of particle trajectories occurs. This is because in 1D ZA describes collapse of parallel sheets of matter and acceleration of each sheet is independent of \mathbf{x} until sheets cross.
- Although ZA is only an approximation in 3D, it still works very well at the initial stages of evolution of cosmological density fields because flattened *pancake* structures (whose evolution is well described by ZA) are typical in such fields⁹ [see, e.g., [Bond et al. \(1996\)](#)]. This makes ZA the method of choice for setting up initial conditions in cosmological simulations.

⁹Another explanation of the success of ZA is its use of displacement field, $S(q)$, as the basis for evolution model. Density depends on the derivatives of $S(q)$ so that small (linear) displacements can correspond to large density contrasts.

A test problem: 1D collapse of a plane wave

- Let's consider the evolution of a 1D sine-wave $S(q) = A \sin(kq)$:

$$x(a) = q + D_+(a)A \sin(kq); \quad k = 2\pi/\lambda;$$

- Let's assume we want to simulate a wave with wavelength equal to the simulation box size, $\lambda = L_{\text{box}} = N_g$ using $N_{p,1}^3$ particles and N_g^3 grid cells in a cubic grid. In 3D, the 1D equations can be used to set up initial conditions for $N_{p,1}$ parallel sheets of particles. For some initial epoch, a_{ini} , we have

$$x_i(a_{\text{ini}}) = q_i + D_+(a_{\text{ini}})A \sin\left(\frac{2\pi q_i}{L_{\text{box}}}\right); \quad q_i = (i-1)\frac{L_{\text{box}}}{N_{p,1}} \text{ for } i = 1, \dots, N_{p,1};$$

- Take the time derivative of x to get equation for *peculiar* velocity $v = a\dot{x}$ (or momentum $p = av$, if needed):

$$v(x) = a\dot{D}_+(a_{\text{ini}} - \Delta a/2)A \sin(kq),$$

where initialization for the leapfrog scheme is assumed.

- We can choose the first crossing epoch, say $a_{\text{cross}} = 10a_{\text{ini}}$. The epoch of the first crossing can then be identified as the epoch of the caustic formation (i.e., $\rho(x_{\text{cross}}, a_{\text{cross}}) = \infty$, where $x_{\text{cross}} = q_{\text{cross}} = L_{\text{box}}/2$), which gives us the wave amplitude A :

$$\rho(x, a) = \frac{\bar{\rho}}{[1 + D_+(a) \times Ak \cos(kq)]}, \quad \Rightarrow A = - (D_+(a_{\text{cross}})k)^{-1}.$$

- Positions $x(q, a)$ and velocities $v(q, a)$ is all we need to set up initial conditions for particles.
- The subsequent 1D evolution using the PM code can be tested by comparing x and v given by the above equations to the coordinates and velocities of particles at different epochs using outputs of your code (for example, you can plot phase diagrams, i.e. particles in $v - x$ or $v - q$ plane).

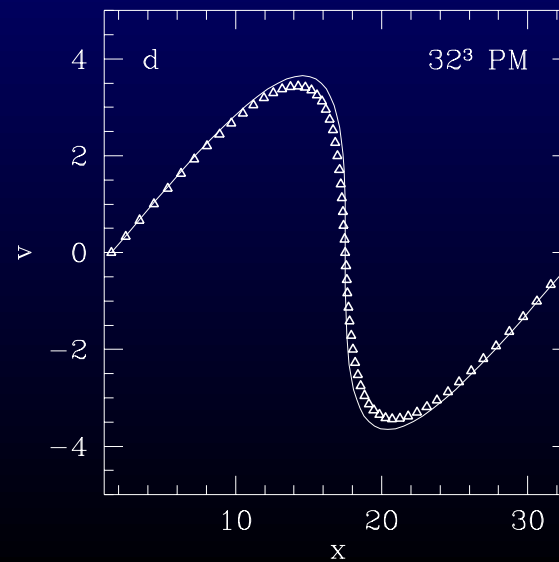
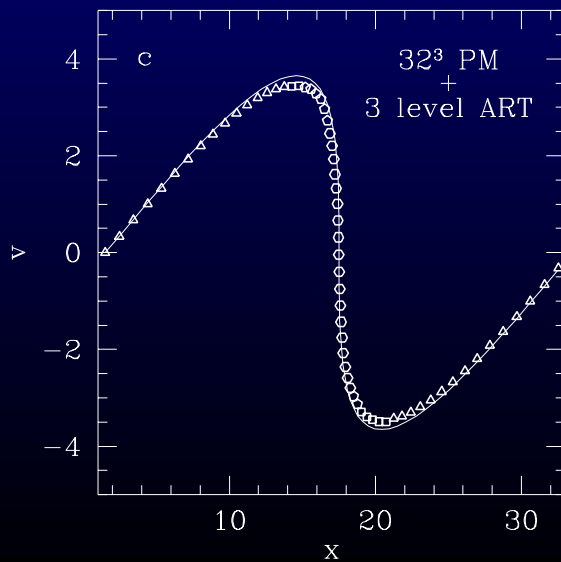
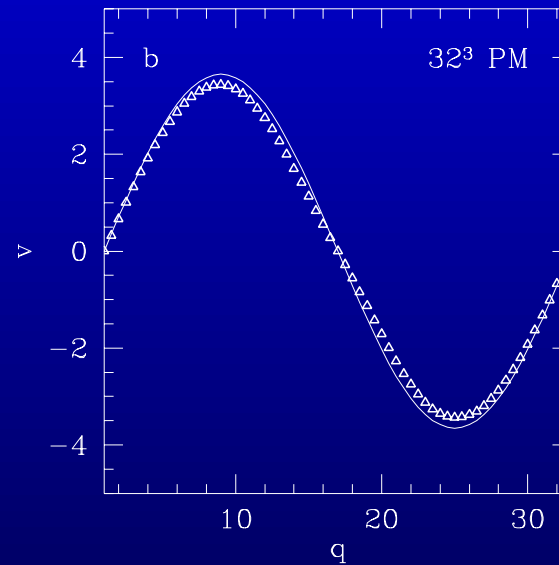
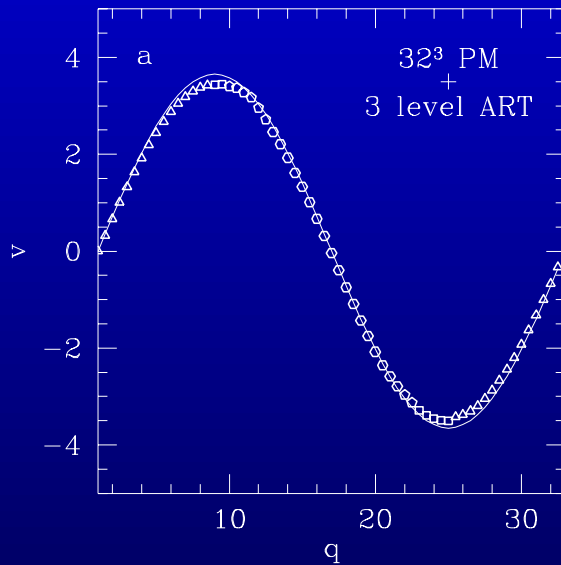
- In addition, you can use solutions for the gravitational potential and particle accelerations to test your gravity solver. For each grid cell:

$$\phi(x, a) = \frac{3\Omega_0}{2a} \left\{ \frac{q^2 - x^2}{2} + D_+ \times Ak [kq \sin(kq) + \cos(kq) - 1] \right\};$$

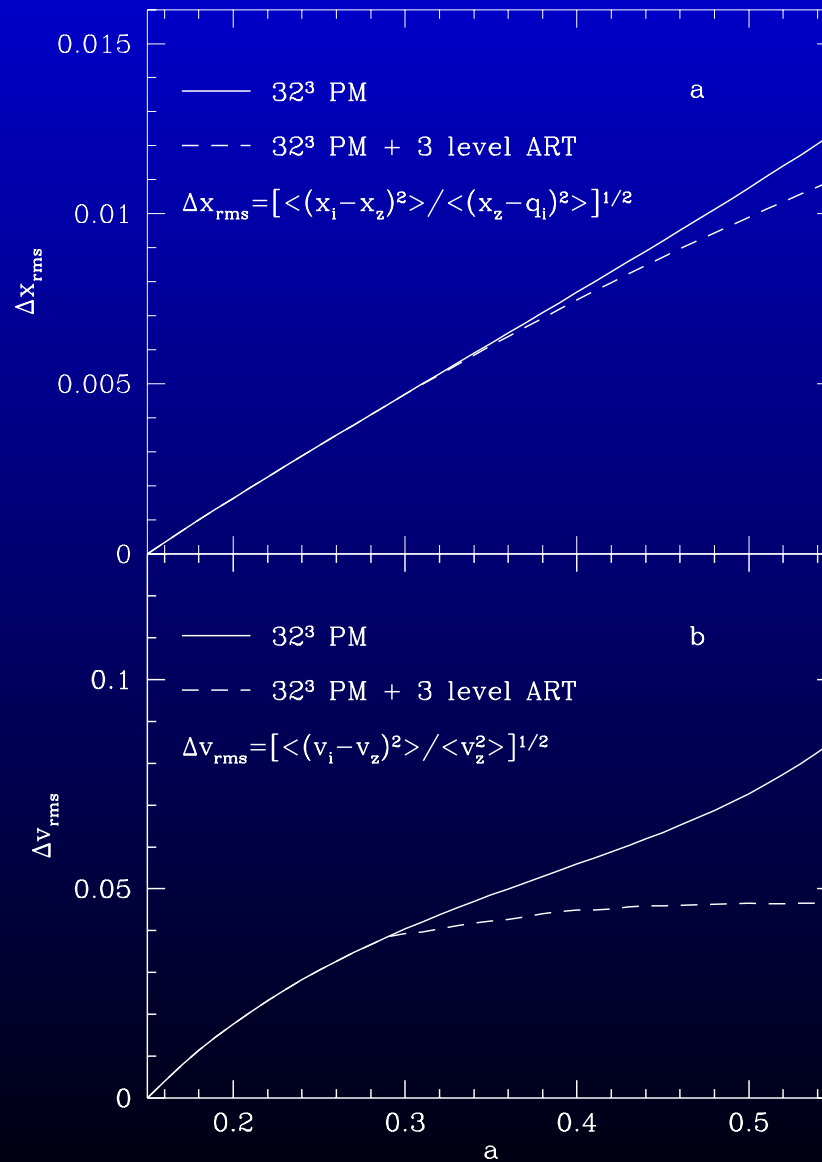
$$g(x, a) = -\frac{\partial\phi}{\partial x} = \frac{3\Omega_0}{2a}(x - q) = \frac{3\Omega_0}{2a}D_+(a)A \sin(kq)$$

The only difference from particles is that you do not know cell's Lagrangian coordinate q (q is known for particles if you maintain particles in the same order throughout evolution; particle's index gives q as we saw above). For a given expansion factor a , q of a cell can be calculated by solving equation $q = x - D_+(a)A \sin(kq)$ numerically, with x is coordinate of the cell.

Plane wave collapse test: phase diagram in Lagrangian coordinates q (a) the ART simulation with 32^3 base grid and 3 refinement levels and (b) the PM simulation with a 32^3 -cell grid at the crossing time. *Solid line*, analytic solution; *polygons*, numerical results. (c,d) Corresponding phase diagram for physical coordinates x .



Plane wave collapse test: rms deviations of (a) coordinates and (b) velocities from analytic solution vs. the expansion parameter for the PM code (*solid line*) and for the ART code with three levels of refinement (*dashed line*) See, [Efstathiou et al. \(1985\)](#) for details on this and other tests.



Beyond the Zeldovich test: setting up cosmological initial conditions

If your code passes the Zeldovich test, you may try to run a realistic cosmological simulation. To do this, you will have to code a routine to set up initial conditions for the particles using a statistical realization of the power spectrum, $P(k)$, of your favorite cosmological model¹⁰. Fortunately, the basis for the algorithm is the now familiar ZA.

The displacement of a particle is now determined not by a single wave, but by the entire set of waves that can be represented numerically in the simulation box. Thus, particle's comoving coordinates and momenta, $p = a^2\dot{x}$, are given by

$$\mathbf{x} = \mathbf{q} - D_+(a)\mathbf{S}(\mathbf{q}); \quad \mathbf{p} = -(a - \Delta a/2)^2 \dot{D}_+(a - \Delta a/2)\mathbf{S}(\mathbf{q}),$$

where a is the initial expansion factor and Δa is its step¹¹, \mathbf{q} is particle's unperturbed position.

¹⁰An alternative is to set up initial conditions using a public code.

¹¹Note that the growth factor $D_+(a)$ is usually scale-independent (e.g., for all models with CDM only) and this is assumed here. This is not true, however, for some models, such as the Cold+Hot Dark Matter (CHDM).

The *displacement vector* \mathbf{S} is given by the discrete Fourier transform (e.g., Padmanabhan 1993, p.294):

$$\mathbf{S}(\mathbf{q}) = \alpha \sum_{k_{x,y,z}=-k_{max}}^{k_{max}} i\mathbf{k} c_k \exp(i\mathbf{k} \cdot \mathbf{q});$$

$$k_{x,y,z} = \frac{2\pi}{N_g} l, m, n; \quad l, m, n = 0, \pm 1, \dots, \pm N_{p,1}/2; \quad k^2 = k_x^2 + k_y^2 + k_z^2 \neq 0.$$

Here, α is the power spectrum normalization. The summation is over all possible wavenumbers from the fundamental mode with wavelength equal to the box size to the smallest “Nyquist” wavelength with the wavenumber of $N_{p,1}/2$. The real and imaginary components of the Fourier coefficients, $c_k = (a_k - ib_k)/2$ are independent gaussian random numbers with the mean zero and dispersion $\sigma^2 = P(k)/k^4$:

$$a_k = \sqrt{P(k)} \frac{\text{Gauss}(0, 1)}{k^2}, \quad b_k = \sqrt{P(k)} \frac{\text{Gauss}(0, 1)}{k^2}.$$

Note that c_k should satisfy condition $c_k = c_{-k}^* = (a_k - ib_k)/2$.

Thus, if we construct a realization of $\{k_{x,y,z}c_k\}$ using a cosmological power spectrum on a grid in the Fourier space, its discrete FFT gives us components of the real space displacement vector $S(\mathbf{q})$ for all particles.

Applying these equations in practice for N_p particles on a cubic grid with N_g^3 cells amounts to

1) distributing particles uniformly in the simulation volume (e.g., placing particles in the centers of appropriately spaced grid cells).

2) Computing the displacement vector for each particle position. Construct a realization of $k_{x,y,z}c_k$ on three (each for one component of the wavevector) cubic grids. For example, for the x -component, the grid is initialized to $k_x c_k$ where k_x are running from $-k_{Ny}$ to k_{Ny} (assume $k_{Ny} = N_{p,1}/2$, k here is in units of the fundamental mode $2\pi/L_{box} = 2\pi/N_g^{1/3}$) and a_k and b_k are gaussian random numbers defined above.

3) FFT each of the three grids to get $S_x(q)$, $S_y(q)$, and $S_z(q)$ in real space. Apply ZA to displace particles from their Lagrangian positions ($\mathbf{q} \rightarrow \mathbf{x}$).

You will need

- Your favorite FFT solver, if you don't have one see *Numerical Recipes* (Ch. 12).
- A decent random number generator. Be careful here as you will need to generate fairly long sequences of random numbers (I can supply you with a RNG, if needed). The gaussian random pairs can be generated from the uniformly distributed numbers using the Box-Muller method (*Numerical Recipes, Ch. 7*).
- A routine returning $P(k)$ for a specific cosmological model. See [Hu & Sugiyama 1996](#) and [Eisenstein & Hu 1999](#) for useful analytical approximations to $P(k)$.

Project summary

- use the blueprint given in this lecture to code up your own code (C or fortran). Use FFT routine given to you by Anatoly Klypin last week or a routine with more transparent packing (e.g., FFT routine from Ch.12 of “Numerical recipes”)
- Test the code by simulating a sine-wave and comparing it to the Zeldovich solution
- Set up an initial conditions for $64^3/128^3$ or $128^3/256^3$ particles/grid cells and run PM simulations in different cosmologies using the same random seed (which should give you the same phases and the same structures in these cosmologies):
 - ★ $\Omega_0 = 1$ CDM universe with $\sigma_8 = 0.8$. Evolution of the growth factor in this model is simple: $D_+(a) = (a/a_0)$. You can check D_+ by computing the $P(k)$ or two-point correlation function of DM particles at different epochs and comparing it to the expected evolution in linear regime.
 - ★ you can then simulate open CDM ($\Omega_0 = 0.3$) and flat Λ CDM ($\Omega_0 = 0.3$, $\Omega_\Lambda = 0.7$) with the same $\sigma_8(z = 0) = 0.8$ as in the SCDM model above.

- you can do various analyses of these simulations:
 - ★ plot particle distribution in slices
 - ★ plot $P(k)$ at different representative stages of evolution
 - ★ run a halo finder (use yours or I can provide you with one) at different epochs and compare evolution of the mass function in these cosmologies.
 - ★ use your imagination! (e.g., parallelize your PM code with OpenMP)

PM in cosmology: some historical references

- Efstathiou G. & Eastwood J. 1981, MNRAS **194**, 503
- Klypin A.A. & Shandarin S.F. 1983, MNRAS **204**, 891
- Centrella J. & Melott A.L. 1983, Nature **305**, 196-198
- Miller R.H., 1983, ApJ **270**, 390-409
- Efstathiou G., Davis M., White S.D.M., & Frenk C.S. 1985, ApJS **57**, 241-260

Papers with useful info on PM

- Hockney, R. W., and Eastwood, J. W. 1981, “*Computer Simulation Using Particles*”, McGraw-Hill, New York
- Klypin A.A. & Shandarin S.F. 1983, MNRAS **204**, 891
- Efsthathiou G., Davis M., White S.D.M., & Frenk C.S. 1985, ApJS **57**, 241-260
[Review and comparison of cosmological N -body methods]
- Sellwood J.A. 1987, ARA&A **25**, 151
[Review of particle simulation methods]
- Description of Hugh Couchman’s AP³M code
- Bertschinger E. 1998, ARA&A **36**, 599
[The most recent review of numerical techniques used in cosmological simulations and algorithms for setting up the initial conditions.]

- Michael Gross's, PhD Thesis
[useful descriptions of PM and an algorithm for setting up ICs in Ch. 2 and Appendix A]
- Klypin, A. & Holtzman, J. 1997, astro-ph/9712217
"Particle-Mesh code for cosmological simulations"

Web links

- Amara's Recap of Particle Simulation Methods: collection of info and links on various N -body algorithms
- Anatoly Klypin's public PM package for cosmological simulations.
- Useful `fortran` (requires `PGPLOT library`) and `C` (requires OpenGL) codes for visualization of particle distributions in 3D.